Toolbox Scilab : Detection signal design for failure detection and isolation for linear dynamic systems User's Guide

Hector E. Rubio Scola

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

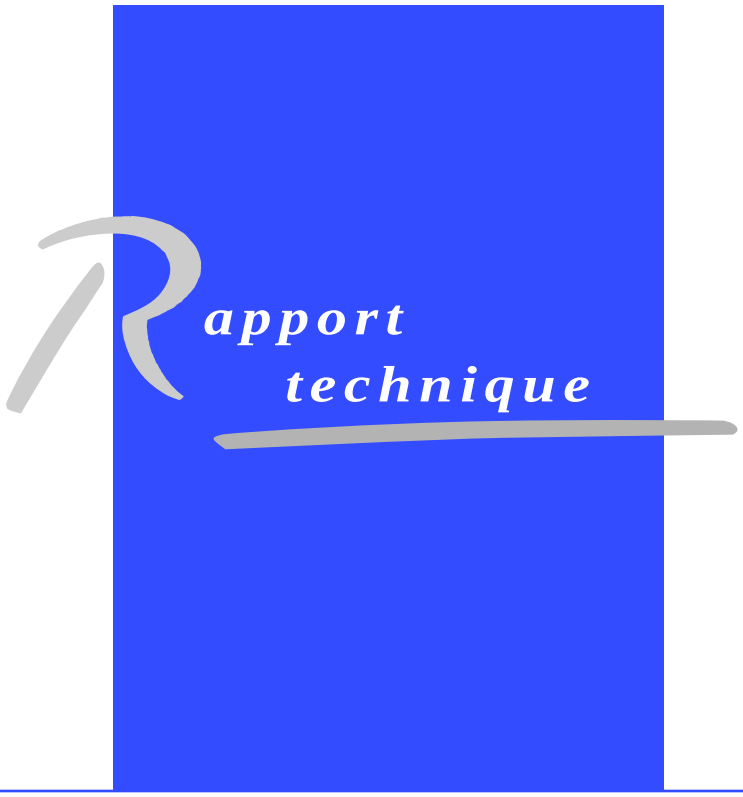
Toolbox Scilab: Detection signal design for failure detection and isolation for linear dynamic systems User's Guide

Héctor E. Rubio Scola

No 0241

Juillet 2000

THÈME 4



ISSN 0249-0803

Toolbox Scilab: Detection signal design for failure detection and isolation for linear dynamic systems
User's Guide

Héctor E. Rubio Scola

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Metalau

Rapport technique n˚0241 — Juillet 2000 — 16 pages

Abstract: We show how the Detection Toolbox can be used in Scilab. We present a Toolbox to calculate a filter for failure detection and isolation in dynamic systems. This Toolbox uses Scilab and the LIPSOL library (Linear programming Interior Point SOLvers).

Key-words: failure detection, large scale programming, failure isolation, bounded perturbations, active detection.

(Résumé : tsvp)

CIUNR - FCEIA - Universidad Nacional de Rosario, Argentina

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Toolbox Scilab: Conception de signal de détection et l'isolation de panne pour les systèmes dynamiques linéaires
Guide d'utilisation

Résumé : Nous montrons comment la Toolbox Détection peut être utilisée dans Scilab. Nous présentons une Toolbox pour calculer un filtre pour la détection et l'isolation de panne dans les systèmes dynamiques. Cette toolbox utilise Scilab et la bibliothèque de LIPSOL.

Mots-clé : détection de panne, programmation grande échelle, isolation de panne, perturbations bornées, détection active

Contents

1 Introduction

This report describes a user-friendly Scilab package, to design a filter for failure detection and isolation in dynamic systems. This Toolbox uses Scilab and the LIPSOL library (Linear programming Interior Point SOLvers).

The detection signal is any input signal that improves the detectability property of failure in a dynamic process. Assuming that the normal and the failed behaviors of a process can be modeled by two linear systems subject to inequality bounded perturbations, a method for on line implementation of a detection signal is used. This method is based on linear programming techniques.

There are two ways of tackling the problem of failure detection and isolation. In first place, a passive approach: the detector monitors the inputs and the outputs of a system to know whether a failure has occurred and, if possible, what kind of failure. To achieve this, the measured input-output is compared with the normal behavior of the system.

The active approach to failure detection consists in acting upon the system on a periodic basis or at critical times using a test signal so as to show abnormal behaviors which would otherwise remain undetected during normal operation (Nikoukhah,1998).

The detector can act either by taking over the usual inputs of the system or through a special input channel, perhaps modifying the structure of the system. The structure of the failure detection method considered in this rapport is depicted in Figure 1.

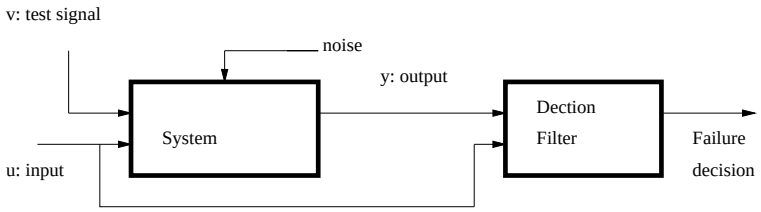


Figure 1: Active failure detection.

A structure of the failure detection and isolation method is proposed in this work at some given time during normal operation of the system, a test signal is injected into the system for a finite period of time. This signal is supposed to expose the failure modes of the system which are then detected by the detection filter.

The problem considered here has two parts:

Part 1: Find a signal v such that the possible input-output set of the normal system be disjoint to the possible input-output set of the failed system.

Part 2: For a found test signal v, given a input-output, recognize if this input-output is in the set of the normal system or in the set of the failed system.

2 System model

We shall consider systems which can be modeled as follows:

$$\begin{array}{rcl} x_s(k+1) & = & A_s(k)x_s(k) + B_s(k)u(k) + b_s(k) + M_s(k)\nu_s(k) \\ y(k) & = & C_s(k)x_s(k) + D_s(k)u(k) + d_s(k) + N_s(k)\nu_s(k) \end{array} \quad (1)$$

for $k = 0, ..., N-1$, where $v = v(k), k \in [0, N-1]$ is a test signal , u and y are inputs and outputs measured on-line. A_s, B_s, C_s, D_s, M_s, N_s, b_s and d_s are matrices and vectors of appropriate dimensions which depend on v and we use the notation $A_s(k) = A_s(v(k))$, etc.
The (known) inputs $u(k)$ and the (unknown) perturbations $\nu_i(k)$ are both supposed to satisfy

$$\begin{array}{rcl} R_{\nu s}(k)\nu_s(k) & \leq & p_{\nu s}(k) \\ R_{us}(k)u(k) & \leq & p_{us}(k) \end{array} \quad (2)$$

where $x_s(k) \in \Re^{n(k)}$, $u(k) \in \Re^{m(k)}$, $y(k) \in \Re^{l(k)}$, $\nu_s(k) \in \Re^{h(k)}$, $p_{\nu s}(k) \in \Re^{q(k)}$, $p_{us}(k) \in \Re^{j(k)}$, and $R_{\nu s}(k)$, $R_{us}(k)$ are given matrices of appropriate dimensions. The vectors $p_{\nu s}(k)$, $p_{us}(k)$ and the matrices $R_{\nu s}(k)$, $R_{us}(k)$ depend on v. No assumption is made on $R_{\nu s}$, and R_{us} except that the inequations (2) are consistent.

In the same way, for the failed system, we choose a similar model in the variables x_f, u, y, ν_f, consider the system with failure as follows:

$$\begin{array}{rcl} x_f(k+1) & = & A_f(k)x_f(k) + B_f(k)u(k) + b_f(k) + M_f(k)\nu_f(k) \\ y(k) & = & C_f(k)x_f(k) + D_f(k)u(k) + d_f(k) + N_f(k)\nu_f(k) \end{array} \quad (3)$$

for $k = 0, ..., N-1$,

Perturbation ν_f and input u satisfy the inequality constraints:

$$\begin{array}{rcl} R_{\nu f}(k)\nu_F(k) & \leq & p_{\nu f}(k) \\ R_{uf}(k)u(k) & \leq & p_{uf}(k) \end{array} \quad (4)$$

where $x_f(k) \in \Re^{n_f(k)}$, $u(k) \in \Re^{m(k)}$, $y(k) \in \Re^{l(k)}$, $\nu_f(k) \in \Re^{h_f(k)}$, $p_{\nu f}(k) \in \Re^{q_f(k)}$, $p_{uf}(k) \in \Re^{j_f(k)}$ and $R_{\nu f}(k)$, $R_{uf}(k)$, are given matrices of appropriate dimensions which depend on v. The vectors $p_{\nu f}(k)$, $p_{uf}(k)$ and the matrices $R_{\nu f}(k)$, $R_{uf}(k)$ depend on v. No assumption is made on $R_{\nu f}$, R_{uf} except that (4) are consistent, for $k = 0, ..., N-1$.

The states of the models x_s and x_f need not have the same dimensions. The same is true for ν_s and ν_f. The systems have in common only the input $u(k)$ and the output $y(k)$.

The basic assumption for failure detection and isolation is that the normal mode, and the failed mode of the system can be modeled as in (1), (2), (3) and (4). But the system matrices can (and hopefully are) different for different modes. Note that unlike most other approaches to uncertainty modeling in dynamical systems for the purpose of failure detection, ν is not a stochastic white noise sequence, but rather an inequality bounded arbitrary discrete sequence.

A fundamental, and reasonable, assumption here is that, during the test period, the system is either in normal mode or failed mode; no transition occurs during the test period.

3 Detection filter implementation

The construction of a proper test signal is considered in the section 4, here we have assumed that the test signal $v = \{v(k), k \in [0, N-1]\}$ is given.

We introduce the vectors w, defined by

$$w(k) = \begin{cases} \left[x_s(k)^T, x_f(k)^T, y(k)^T, u(k)^T, \nu_s(k)^T, \nu_f(k)^T\right]^T & \text{if} \quad k \in [0, N-1] \\ \left[x_s(N)^T, x_f(N)^T\right]^T & \text{if} \qquad k = N \end{cases}$$

(1), (2), (3) and (4) can be rewritten as:

$$\begin{array}{lcl} F_1 & w & = & p \\ F_2 & w & \leq & q \end{array} \tag{5}$$

We write in matrix form the equations and inequations for normal system (1), (2) and failed system (3), (4) separately, for the normal system :

$$\begin{array}{lcl} F_{s,1} & w & = & p_s \\ F_{s,2} & w & \leq & q_s \end{array} \tag{6}$$

and for the failed system :

$$\begin{array}{lcl} F_{f,1} & w & = & p_f \\ F_{f,2} & w & \leq & q_f \end{array} \tag{7}$$

We define the following polyhedrons :

$$S_s = \{w \ / \ w \text{ verifies } (6)\}, \quad S_f = \{w \ / \ w \text{ verifies } (7)\}.$$

Given a signal v if the solution sets of systems (6) and (7) have no intersection, i.e. $S_s \cap S_f = \emptyset$, then system (5) has no solution. In this case we say that v is a proper detection signal.

Let us now introduce new variables (y_s, u_s) as an normal system output-input pair in (1), (2) and (y_f, u_f) as a failed system output-input pair in (3), (4). We also introduce the deviation between input-output pairs for normal and failed systems as follows :

$$e(k) = \begin{bmatrix} e_y(k) \\ e_u(k) \end{bmatrix} = \begin{bmatrix} y_s(k) \\ u_s(k) \end{bmatrix} - \begin{bmatrix} y_f(k) \\ u_f(k) \end{bmatrix} \tag{8}$$

Let $\bar{w} = [w^T, e^T]^T$ where w is the new solution vector corresponding to (u_s, y_s, u_f, y_f) in (1), (2), (3) and (4). The set of all equation and inequation satisfied by $\bar{w}$ can be written as :

$$\begin{array}{lcl} [F_1 \quad E_1]\bar{w} & = & p \\ [F_2 \quad E_2]\bar{w} & \leq & q \end{array} \tag{9}$$

Clearly, (5) is equivalent to (9) with additional constraint $e = 0$, so that (5) is inconsistent (i.e. v is a proper detection signal) if and only if (9) has no solution satisfying $e = 0$. The projection of all the $\bar{w}$' s satisfying (9) on the e coordinate is the same as projection of $S_s - S_f$ on the (u, y) coordinate.

Given $w = w(k), k \in [0, N]$ if the vector w satisfies (6) there is no fault and if it satisfies (7) there is fault.

Since inequalities (6) and (7) define two disjoint convex polyhedrons, our problem is reduced to know in which polyhedron a given vector w lives. If a hyperplane can be found that separates the polyhedrons, it is sufficient to find at which side of the hyperplane lies the vector w. Our work is limited to find that hyperplane. Its existence is guaranteed by the classical convexity theory. We can thus obtain directly constraints involving only inputs u and outputs y for testing if there is failure. Therefore we do not need to know the states variables x_s and x_f of the systems (1) and (3). Then the failure test is reduced to one single inequality test :

$$h_u u + h_y y < d \quad \text{or} \quad h_u u + h_y y > d$$

where

$$h_u u + h_y y = d \tag{10}$$

is the equation of separating hyperplane. The h_u, h_y coefficients are calculated by the methodology described in ([3]) and so is calculated the d term.

$$u = [u(0), u(1), ..., u(N-1)], \quad y = [y(0), y(1), ..., y(N-1)].$$

3.1 Scilab functions

To calculate h_u, h_y of (10) we use the `hiper` function and to calculate the d term of (10) we use the `bound` function. To do that, we need to use two utility functions.

3.1.1 Utility functions

In this paragraph, we describe the utility functions. The function `sys2tlist` does a `sys` list with the normal system matrices and vectors of (1) and (2), and a `sys1` list with the failed system matrices and vectors of (3) and (4). The function `matFE` forms the matrices and vectors of the equations (5) (6), (7) and (9).

tlist2sys - sys2tlist functions

Syntax

```
[A, B, C , D, b, M, d, N, Ru, pu, Rv, pv] = tlist2sys(sys)
[sys] = sys2tlist (A, B, C ,D, b, M, d, N, Ru, pu, Rv, pv)
```

Description

Utility functions, the `sys2tlist` function define a linear system as a tlist `sys`. Inversely the `tlist2sys` functions returns the `A, B, C ,D, b, M, d, N, Ru, pu, Rv, pv` matrices or function from a linear system `sys`.

$$\begin{array}{rcl} x(k+1) & = & A(k)*x(k)+B(k)*u(k)+b(k)+M(k)*w(k) \\ y(k) & = & C(k)*x(k)+D(k)*u(k)+d(k)+N(k)*w(k) \\ & & Ru(k)*u(k) < pu(k) \\ & & Rv(k)*w(k) < pv(k) \end{array}$$

We have used the notations `A(k) = A(v(k))`, etc... Note: `A, B, C,` etc. are real matrices or functions. If `A` is a matrix then all the other elements of the system must be matrices. Similarly if `A` is a function then all the other elements of the system must be functions.

matFE function

Syntax

```
[F,E,RHS,mi,index]=matFE(sys,sys1,vp,F0,RHS0,mi0)
```

Description

The function `matFE` forms the equations (5), (6), (7) and (9). The matrices and vectors of equation (9) are :

$F_1 =$ `F(1:mi,:)` $\quad E_1 =$ `E(1:mi,:)` $\quad p =$ `RHS(1:mi)`
$F_2 =$ `F(mi+1:$,:)` $\quad E_2 =$ `E(mi+1:$,:)` $\quad q =$ `RHS(mi+1:$)`

With the `index` list of the vector we obtain (6) and (7),
`[index(1); index(3)]` are the rows corresponding to the normal system,
`[index(2); index(4)]` are the rows corresponding to the failed system,
`index(5)` are the columns corresponding to input u(k) and the output y(k),
`index(6)` are the columns corresponding to the normal system and
`index(7)` are the columns corresponding to the failed system.
The matrices and vectors of equations (6) and (7) are :

$F_{s,1} =$ `F(index(1),index(6))` $\quad p_s =$ `RHS(index(1))`
$F_{s,2} =$ `F(index(3),index(6))` $\quad q_s =$ `RHS(index(3))`
$F_{f,1} =$ `F(index(2),index(7))` $\quad p_f =$ `RHS(index(2))`
$F_{f,2} =$ `F(index(4),index(7))` $\quad q_f =$ `RHS(index(4))`

The `sys` and `sys1` tlist are formed with `sys2tlist` function and the `vp` matrix is test signal. The initial state `x(0)` is defined by `F0*x(0)` $\leq$ `RHS0` and `mi0` is the number of equality constraints (i.e. `F0(1:mi,:) * x(0) = RHS0 (1:mi0)`).

3.1.2 Main functions

hiper function

Syntax

```
[bb] = hiper (vp,F,E,RHS,mi [,cc])
```

Description

The function hiper give a feasible direction `bb` $= [h_u, h_y]$ for a separating hyperplane. If `bb` is a matrix, then $[h_u, h_y]$ can be a column of `bb`.

```
bb * [y(0);u(0);y(1);u(1); ....] = d
```

To calculate `d` see function `bounds`. The `vp` matrix is the test signal. The `F, E, RHS, mi` matrices are obtained with the function `matFE`. If not given, the bound variables: `cc` take their default values `cc = [-10^3,10^3]`.

bounds function

Syntax

```
[d, bs] = bounds(F, RHS, mi,index, bb [,cc])
```

Description

The `F`, `E`, `RHS` `mi` and `index` are obtained by the `matFE` function, the `bb` is obtained by the `hiper` function.

The `bounds` function calculates the values of the d term of the hyperplane equation that are bounded by `bs(1)` and `bs(4)` or `bs(2)` and `bs(3)`

$$\begin{array}{ll} \texttt{bs(1)} = \min\limits_{s.t.7} \texttt{bb*[y,u]} & \texttt{bs(2)} = \max\limits_{s.t.7} \texttt{bb*[y,u]} \\ \texttt{bs(3)} = \min\limits_{s.t.6} \texttt{bb*[y,u]} & \texttt{bs(4)} = \max\limits_{s.t.6} \texttt{bb*[y,u]} \end{array} \tag{11}$$

If `bs(1)`< `bs(4)` then `d` = (`bs(2)` + `bs(3))/2`, or if `bs(2)`> `bs(3)` then `d` = (`bs(1)` + `bs(4))/2`. Then the hyperplane equation will be :

```
bb*[y(0), u(0), y(1), u(1), ..., y(N-1), u(N-1)] = d
```

If `d = []` then there is a warning message concerning with `cc` bounds. If not given, the bound variables: `cc` take their default values `cc = [ -10^3, 10^3]`.

3.2 Example

The results are taken from random systems. The normal system has four state variables and one output. The failed system has five state variables and one output.

To solve this problem using `hiper`, we first need to construct tlist for two systems. To do that we use the `sys2tlist` function and then with `matFE` we form the necessary matrices.

```
rand('seed',0);
A=rand(4,4)/2;b=rand(4,1);M=[eye(4,4),zeros(4,1)];
C=rand(1,4);d=rand(1,1);N=[zeros(1,4),1];
Rv= [eye(5,5);-eye(5,5)]; pv =  0.01*ones(10,1);
Ru = []; pu=[];B=[];D=[];
[sys]=sys2tlist(A,B,C,D,b,M,d,N,Ru,pu,Rv,pv); // normal system

A=rand(5,5)/3;b=rand(5,1);M=[eye(5,4),zeros(5,1)];
C=rand(1,5);d=rand(1,1);N=[zeros(1,4),1];
Rv = [eye(5,5);-eye(5,5)]; pv =  0.01*ones(10,1);
Ru =[];pu=[];B=[];D=[];
[sys1]=sys2tlist(A,B,C,D,b,M,d,N,Ru,pu,Rv,pv); // failed system

RHS0=[];F0=[];mi0=0; // the initial state
```

`RHS0=[]` and `F0=[]` indicate that there are not conditions for the initial state. The simple test signal would be

```
vp=[-0.9 -0.9 -0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 -0.9];
// the test signal
```

With the `matFE` function we obtain the matrices `F` and `E`, the vector `RHS` and list `index` with necessary index.

```
[F,E,RHS,mi,index]=matFE(sys,sys1,vp,F0,RHS0,mi0);
```

The `hiper` function gives a feasible direction `bb` for a separating hyperplane.

```
cc = [-10^4,10^4];[bb] = hiper (vp,F,E,RHS,mi,cc)
```

Once `bb` is calculated, it remains to select among the family of parallel hyperplanes, a separating hyperplane, i.e. we have to choose `d` in the hyperplane equation.

This is done by looking for hyperplanes tangent to the polyhedrons defined by : $-0.0026103y(1) - 0.0124951y(2) - 0.1134797y(3) - 0.8806846y(4) - 0.6068333y(5) - 11.137272y(6) + 12.152271y(7) + 9.7878951y(8)0.1405349y(9) + 2.4110049y(10) + 7.6629585y(11) - 21.044658y(13) = d$

We can solve the problem in Scilab as follows

```
cc = [-10^5,10^4];[d,bs]=bounds(F,RHS,mi,index,bb,cc),
```

We obtain

```
 bs  =
! - 29.035421 !
! - 24.332146 !
! - 1.0847342 !
!   14.965766 !
  d =
    - 12.70844
```

If $-29.035 \leq \mathtt{d} \leq -24.332$, the system is normal, if $-1.084 \leq \mathtt{d} \leq 14.965766$, the system has failed. Then the hyperplane equation is :

$-0.0026103y(1) - 0.0124951y(2) - 0.1134797y(3) - 0.8806846y(4) - 0.6068333y(5) - 11.137272y(6) + 12.152271y(7) + 9.7878951y(8) 0.1405349y(9) + 2.4110049y(10) + 7.6629585y(11) - 21.044658y(13) = 14.965$

The following figures shows dynamic responses of systems. The curve A is for the normal system and curve B for the failed systems.

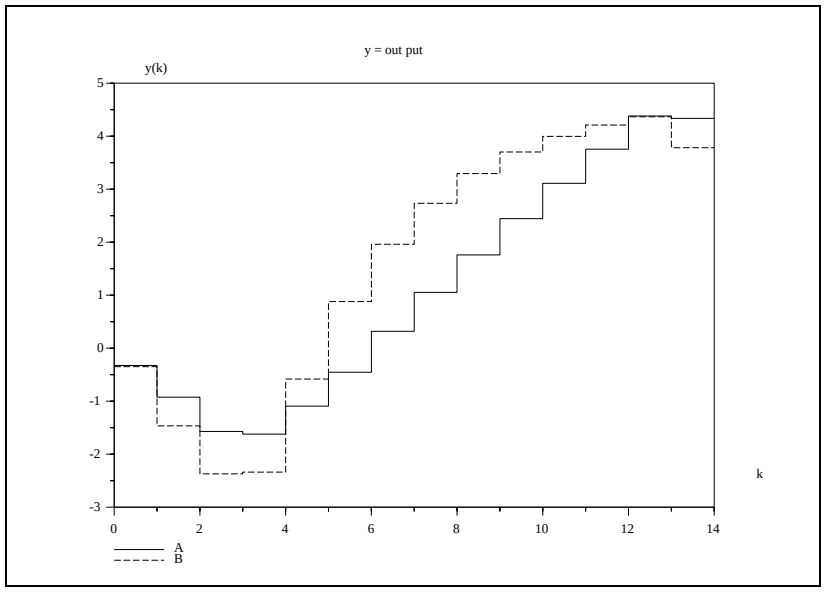


Figure 2: Output y

```
[yy,yy1]=simul(sys,sys1,vp,[]);
xbasc();N = size(yy,2);k = 0:1:N;
plot2d2('onn',k',[[yy(1,:),yy(1,$)];[yy1(1,:),yy1(1,$)]]', ...
[1,2],"161","A@B"  );xtitle('y = out put ','k', 'y(k)');
```

4 Test signal design

In this section, we show how we can find a detection horizon N and construct a test signal $v = \{v(k), k \in [0, N-1]\}$, as short as possible, such that (1), (2) and (3), (4) are mutually exclusive, i.e. $S_f \cap S_s = \emptyset$.

The solution to this problem is only given in the case where the test signal enters the system linearly. We show how a test signal can be designed for a special class of model (1), (2), (3) and (4). We assume that the matrices $A_i(k)$, $B_i(k)$, $C_i(k)$, $D_i(k)$, $M_i(k)$ $N_i(k)$, $R_{\nu f}(k)$, $R_{uf}(k)$ and the vectors $p_{\nu s}(k)$, $p_{us}(k)$ for $i = s, f$ do not depend on v and that

$$\begin{array}{rcl} b_s(v(k)) & = & b_{s,1}(k)v(k) \\ d_s(v(k)) & = & d_{s,1}(k)v(k) \end{array} \tag{12}$$

$$\begin{array}{rcl} b_f(v(k)) & = & b_{f,1}(k)v(k) \\ d_f(v(k)) & = & d_{f,1}(k)v(k) \end{array} \tag{13}$$

where $b_{i,1}(k)$ and $d_{i,1}(k)$ are matrices of appropriate dimensions, for $i = s, f$.

4.1 Main functions

To calculate the test signal there exist two scilab functions with different methodologies: `sigtest` and `sigintest`

sigtest function

Syntax

```
[V, lam]=sigtest(Ver, sys, sys1, F0, RHS0, mi0 [,cc])
```

Description

The `sys` and `sys1` tlist are formed with `sys2tlist` function. The initial state `x(0)` is defined by `F0*x(0)` $\leq$ `RHS0` and `mi0` is the number of equality constraints (i.e. `F0(1:mi,:) * x(0)` = `RHS0 (1:mi0)`). The `Ver` real vector is `[Ver(1);Ver(2)]`.

The `sigtest` function gives `V` (`length(V)` as small as possible). Then we calculate `v` test signal.

`v = V *lam1` with `lam < lam1 < 1`

The criteria for choosing a detection signal `v` is `Ver(1) < v < Ver(2)`. `V` is such that :

$$\texttt{V(k)} = \begin{cases} \texttt{Ver(i)} & \text{if} \quad k \in [0, k_o] \\ \texttt{Ver(j)} & \text{if} \quad k \in [(k_o+1), (N-1)] \end{cases} \tag{14}$$

or

$$\texttt{V(k)} = \begin{cases} \texttt{Ver(i)} & \text{if} \quad k \in [0, k_o] \\ \texttt{Ver(j)} & \text{if} \quad k \in [(k_o+1), k_1] \\ \texttt{Ver(i)} & \text{if} \quad k \in [(k_1+1), (N-1)] \end{cases} \tag{15}$$

with $k_o, k_1 \in [0, N-1]$ and $i \neq j; i = 1, 2; j = 1, 2$. If not given, the bound variables: `c(1)` and `c(2)` take their default values `cc=[-10^3, 10^3]`

sigintest function

Syntax

```
[V, lam]=sigintest(Ver, sys, sys1, F0, RHS0, mi0 [, cc])
```

Description

The `sigintest` function gives `V` using auxiliary systems methode. The test signal `v` is defined by

`v(k+1) = v(k) + V(k)*lam1` with `lam < lam1 < 1`

`V(k)` such that `V(k) = Ver(i)`, with $i = 1$ or $i = 2$ and `k`$= 1, ..., N-1$ such that minimizes the function :

$$\| \sum_{k=0}^{N-1} \texttt{V(k)} \|$$

We can see this problem as a way of aggregating one state variable to our system. The equations (1), (4) are transformed as :

$$\begin{array}{rcl} \begin{bmatrix} x_i(k+1) \\ \texttt{v(k+1)} \end{bmatrix} & = & \begin{bmatrix} A_i & b_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_i(k) \\ \texttt{v(k)} \end{bmatrix} + \begin{bmatrix} B_i \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \texttt{V(k)} + \begin{bmatrix} M_i \\ 0 \end{bmatrix} \nu_i(k) \\ y(k) & = & \begin{bmatrix} C_i & d_i \end{bmatrix} \begin{bmatrix} x_i(k) \\ \texttt{v(k)} \end{bmatrix} + D_i u(k) + N_i \nu_i(k) \end{array} \tag{16}$$

with $i = s, f$

Given `V`, a test signal constructed for the extended systems, the signal `v` for the original systems is completely defined once `v(0)` (or any `v(k)`) is fixed.

The systems `sys` and `sys1` are auxiliary systems, see `integra` function. The initial state of the auxiliary systems `x(0)` is defined by `F0*x(0)` $\leq$ `RHS0` and `mi0` is the number of equality constraints (i.e. `F0(1:mi,:) * x(0) = RHS0 (1:mi0)`). If not given, the bound variables: `c(1)` and `c(2)` take their default values, `cc =[-10^3,10^3]`

4.1.1 Utility functions

integra function

Syntax

```
[sysi] = integra(sys)
```

Description

Utility function: converts the `sys` system that have the equation(1) into the system `sysi` system the equation that have (16).

4.2 Examples

4.2.1 How to use the sigtest function

Consider the systems

```
rand('seed',0)
n=4;
A=rand(n,n);b=rand(n,1);M=[eye(n,n),zeros(n,1)];
C=rand(1,n);d=rand(1,1);N=[zeros(1,n),1];
Ru = []; pu=[];B=[];D=[];
Rv = [eye(n+1,n+1);-eye(n+1,n+1)]; pv = 0.01*ones(2*n+2,1);
[sys]=sys2tlist(A,B,C,D,b,M,d,N,Ru,pu,Rv,pv); // normal system
n1=5;
A=rand(n1,n1);b=rand(n1,1);M=[eye(n1,n),zeros(n1,1)];
C=rand(1,n1);d=rand(1,1);N=[zeros(1,n),1];
Ru = []; pu=[];B=[];D=[];
Rv = [eye(n+1,n+1);-eye(n+1,n+1)]; pv = 0.01*ones(2*n+2,1);
[sys1]=sys2tlist(A,B,C,D,b,M,d,N,Ru,pu,Rv,pv); // failed system
RHS0 =[];F0=[];mi0=0; // the initial state

Ver = [-300 300];
[V,lam]=sigtest(Ver,sys,sys1,F0,RHS0,mi0)
```

We obtain the following result

```
-->V
 V  =
! -300. 300. 300. 300. 300. 300. 300. 300. 300. 300. !

-->lam
 lam  =
  0.0032935
```

The signal test is

```
-->v = V*0.004
 v  =
 ! -1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 1.2 !
```

4.2.2 How to use the sigintest function

```
rand('seed',0);
A=rand(4,4);b=rand(4,1);M=[eye(4,4),zeros(4,1)];
C=rand(1,4);d=rand(1,1);N=[zeros(1,4),1];
Ru = []; pu=[];B=[];D=[];
Rv = [eye(5,5);-eye(5,5)]; pv = 0.1*ones(10,1);
[sys]=sys2tlist(A,B,C,D,b,M,d,N,Ru,pu,Rv,pv);
     // normal system
A=rand(4,4);b=rand(4,1);M=[eye(4,4),zeros(4,1)];
C=rand(1,4);d=rand(1,1);N=[zeros(1,4),1];
Ru = []; pu=[];B=[];D=[];
Rv = [eye(5,5);-eye(5,5)]; pv = 0.1*ones(10,1);
[sys1]=sys2tlist(A,B,C,D,b,M,d,N,Ru,pu,Rv,pv);
// failed  system

[sysi]=integra(sys);       // auxiliary normal  system
[sysi1]=integra(sys1);     // auxiliary failed  system
mi0=1; F0=[0 0 0 0 1]; RHS0= 0;
// the initial condicion of auxiliary state

Ver  = [-300 300]    // criteria for choosing a detection signal v
[V,lam]=sigintest(Ver,sysi,sysi1,F0,RHS0,mi0);

-->V
 V  =
        column 1 to 9
 ! 300. -300. -300. 300. 300. -300. -300. 300. -300. !

-->lam
 lam  =
   0.0259258

v = 0;          // the initial condicion of signal test
for i=1:size(V,2)-1
v(i+1) = v(i)+V(i);end;
v = v'*0.03,            // signal test

 v  =

!   0.    9.    0.  - 9.    0.    9.    0.  - 9.    0. !
```

5 Installing Scilab Detection

The Detection Toolbox can be obtained by anonymous ftp from `ftp.inria.fr` under
`INRIA/Projects/Meta2/Scilab/contrib/Detection/detection.tar.gz`
and decompress it with:

`gunzip detection.tar.gz` and `tar -xvf detection.tar`

in the subdirectory where you want to install the detection main directory. Read and follow the README instructions.

Detection-Toolbox uses Scilab-Lipsol (Linear programming solving), therefore it is necessary to install it as well.

The LIPSOL-Scilab can be obtained by anonymous `ftp.inria.fr` under
`INRIA/Projects/Meta2/Scilab/contrib/lipsol/LIPSOL.tar.gz`

References

[1] Nikoukhah, R. (1998) Guaranteed Active Failure Detection and Isolation for Linear Dynamical Systems. Automatica, Vol 34, No 11, pp. 1345-1358

[2] Rubio Scola, H. (1997) Implementation of Lipsol in Scilab. Rapport INRIA No 0215.

[3] Rubio Scola, H. (2000) Detection signal design for failure detection and isolation for linear dynamic system. Rapport INRIA No 3935.

[4] Zhang, Y. (1995) User's Guide to LIPSOL, Department of Mathematics and Statistics. University of Maryland Baltimore County. USA.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifi que,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399